Learning Tensor-structured Dictionaries with Application to Hyperspectral Image Denoising

Cássio F. Dantas, Jérémy E. Cohen, Rémi Gribonval
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

***Abstract*—Dictionary learning, paired with sparse coding, aims at providing sparse data representations, that can be used for multiple tasks such as denoising or inpainting, as well as dimensionality reduction. However, when working with large data sets, the dictionary obtained by applying unstructured dictionary learning methods may be of considerable size, which poses both memory and computational complexity issues. In this article, we show how a previously proposed structured dictionary learning model, HO-SuKro, can be used to obtain more compact and readily-applicable dictionaries when the targeted data is a collection of multiway arrays. We introduce an efficient alternating optimization learning algorithm, describe important implementation details that have a considerable impact on both algorithmic complexity and actual speed, and showcase the proposed algorithm on a hyperspectral image denoising task.**

***Index Terms*—Dictionary learning, Tensor, Kronecker product, Hyperspectral imaging, Denoising**

I. Problem Statement

Let $\mathcal{Y}$ be a collection of n tensor data $\{\mathcal{Y}_1, \dots, \mathcal{Y}_n\}$. For the sake of simplicity let us suppose $\mathcal{Y}_i$ are three-mode arrays in $\mathbb{R}^{m_1 \times m_2 \times m_3}$ stacked along the fourth mode to form $\mathcal{Y} \in \mathbb{R}^{m_1 \times m_2 \times m_3 \times n}$ –generalization to higher-order tensors is straightforward. We are interested in the following problem, which is classically formulated for vector input data:

$$\underset{D \in \mathcal{S}_D, x_i}{\operatorname{argmin}} \sum_{i=1}^{n} \| \operatorname{vec}(\mathcal{Y}_i) - D x_i \|_2^2 + g(x_i) \tag{1}$$

where $D \in \mathbb{R}^{m_1 m_2 m_3 \times d}$ is the dictionary which is often overcomplete ($d \geq m_1 m_2 m_3$) and belongs to a constraint set $\mathcal{S}_D$. Function g is a sparsity inducing penalty and x_i is the sparse vector of coefficients describing the vectorized tensor $\mathcal{Y}_i$ in the set of atoms D. Most often, $\mathcal{S}_D$ is the set of matrices with unit Euclidean norm columns.

This problem is coined as Dictionary Learning [1], and has been extensively studied in signal processing over the last decade. When tensor data is considered, two main drawbacks emerge in this formulation: i) it is completely agnostic to the original multidimensional structure of the data; ii) data sizes m_j may be relatively large (even more so the product $m_1 m_2 m_3$) and both the storage of matrix D and the computation of products such as Dx_i may be cumbersome.

One way to tackle the second issue is to restrict the class $\mathcal{S}_D$ of dictionaries that are sought in problem (1). To also tackle the first mentioned issue, we previously proposed the High-Order Sum of Kronecker model (HO-SuKro) [2], a class of tensor-structured dictionaries which are particularly suited to tensorial input data and can be written as follows:

$$D = \sum_{q=1}^{r} D_{1,q} \boxtimes D_{2,q} \boxtimes D_{3,q} \tag{2}$$

where $\boxtimes$ is the Kronecker product [3], $D_{j,q}$ are matrices in $\mathbb{R}^{m_j \times d_j}$ and $d_1 d_2 d_3 = d$. The unit-norm constraint in D is handled as a post-processing step.

It can be shown that this parametrization corresponds to a low (operator-)rank constraint on the dictionary – seen as a trilinear operator acting on tensors – which appears as a natural structural assumption. To provide some extra intuition, for a fixed q we can see the terms $D_{j,q}$ as linear operators acting independently in the j-th mode of the input data $\mathcal{Y}$. For $r = 1$, it boils down to the classic separable operators, which include for instance the 3-dimensional discrete cosine transform (3D-DCT).

An obvious remark about HO-SuKro is that the number of parameters to represent D using (2), which is $r(m_1 d_1 + m_2 d_2 + m_3 d_3)$, is much smaller than the number of entries $m_1 m_2 m_3 d_1 d_2 d_3$ as long as r is small. Moreover, matrix products with the dictionary can be computed markedly faster.

In this paper, we propose an efficient algorithm to learn one such dictionary from data (Section III) and scrutinize the involved computational complexities as well as how they translate into practical speedups (Section IV). Finally, in Section V, we apply the proposed technique to a Hyperspectral image denoising task and compare its performance to some similar approaches as well as the state-of-the-art.

II. State of the art

Modeling dictionaries in the dictionary learning problem as a Kronecker product of several smaller matrices has become a somewhat standard technique, though typically only the rank one case is considered. Early models studied Kronecker products with two terms [4], and later three or any number of terms [5]–[7]. The choice of this Kronecker structure is seldom justified using tensor algebra, but rather as a trick to reduce the number of parameters and computational complexity.

Sums of Kronecker products have been proposed for estimating covariance matrices [8], [9], with only two terms in the Kronecker products. The closest work to our proposed SuKro [10] and HO-SuKro [2] is the Kronecker representation for matrices introduced in [11], which is meant as a solution

to computing a tensor Singular Value Decomposition for structured tensors, where blocs $D_{j,q}$ are orthogonal. It doesn't, therefore, particularly address the dictionary learning problem.

In what follows, we will show that rather simple algorithms can be designed to estimate matrices $D_{j,q}$. This contrasts with sometimes intricate optimization schemes employed by the references mentioned above. In particular, in our previous work on HO-SuKro [2], we proposed a projected alternating algorithm and, despite the theoretical gains in parameter size and memory requirements for the learned dictionary, the learning process itself was quite time-demanding.

III. Proposed algorithm

The dictionary learning problem in equation (1) is a non-convex problem. However, initially ignoring the normalization constraint, the minimization with respect to D with fixed x_i ($\forall i \in \{1,\dots,n\}$) is a quadratic problem which has a closed-form solution, while estimating x_i with known D is a sparse coding problem, which has been extensively studied over the last two decades [12], [13]. Therefore, a usual way to solve (1) is to alternate between estimating only D and estimating only X. We will refer to each subproblem problem respectively as dictionary update and sparse coding.

Now, using the proposed Kronecker structure on D, the sparse coding problem is formally unchanged. Therefore, one may use greedy heuristics such as Orthogonal Matching Pursuit (OMP) [12] or convex relaxations based algorithms such as FISTA [13]. The Kronecker structure can be exploited to positively impact the running time and complexity of these methods, see Section IV for more details.

Dictionary update, on the other hand, is heavily modified in the HO-SuKro formulation. The partial cost function, with respect to the blocks $\{D_{j,q}\}_{q=1,\dots,r}^{j=1,\dots,3}$, may be written as follows:

$$f(D_{j,q}) = \sum_{i=1}^{n} \| \operatorname{vec}(\mathcal{Y}_i) - \sum_{q=1}^{r} (D_{1,q} \boxtimes D_{2,q} \boxtimes D_{3,q})\, x_i \|_2^2 \quad (3)$$

Of course, one may think of multiple techniques to minimize (3), for instance gradient-based approaches, second order methods, or block-coordinate descent. In fact, because of the similarity of (3) with the Tucker Decomposition problem (see [14] for an overview), a first approach we tried was to minimize (3) in an alternating fashion, fixing all parameters but one elementary block $D_{j,q}$. This leads to a quadratic problem, which can be solved in closed form.

However, in the spirit of Alternating Least Squares algorithms (ALS), it is possible to gather all elementary blocks $D_{j,q}$ for a fixed mode j and rather alternate between only three macro-blocks (one per mode) $\Delta_j = \{D_{j,q}\}_{q\in[1,r]}$, $1 \le j \le 3$, and still have a closed-form solution.

A. Quadratic partial cost function explained

Using, for instance, the first-mode unfolding as in [3] denoted $Y_{[1]} \in \mathbb{R}^{m_1 \times m_2 m_3 n}$ for $\mathcal{Y}$ and $X_{[1]} \in \mathbb{R}^{d_1 \times d_2 d_3 n}$ for $\mathcal{X}$, where $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3 \times n}$ is a tensor built similarly to $\mathcal{Y}$ by stacking all coefficient tensors $\mathcal{X}_i = \operatorname{tens}(x_i) \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, one can rewrite (3) as

$$f(D_{j,q}) = \| Y_{[1]} - \sum_{q=1}^{r} D_{1,q} X_{[1]} (D_{2,q} \boxtimes D_{3,q} \boxtimes I_n)^T \|_F^2 \quad (4)$$

The identity matrix I_n that appears in the Kronecker structure is an implicit way to apply the dictionary to $\mathcal{X}_i$ simultaneously for all $i \in \{1,\dots,n\}$. Equation (4), in turn, can be seen as a matrix factorization problem

$$f(D_{j,q}) = \| Y_{[1]} - \Delta_1 U \|_F^2 \quad (5)$$

where $\Delta_1 = [D_{1,1}, \dots, D_{1,r}] \in \mathbb{R}^{m_1 \times r d_1}$ contains the horizontally-stacked matrices $D_{1,q}$ with $1 \le q \le r$, and $U = [U_1; \dots; U_r] \in \mathbb{R}^{r d_1 \times m_2 m_3 n}$ the vertically-stacked right-hand terms with $U_q = X_{[1]} (D_{2,q} \boxtimes D_{3,q} \boxtimes I_n)^T$.

Note that f is quadratic with respect to Δ_1, and an optimal solution for fixed U is given in closed form by

$$\widehat{\Delta_1} = \operatorname{argmin}_{\Delta} \| Y_{[1]} - \Delta_1 U \|_F^2 = Y_{[1]} U^\dagger. \quad (6)$$

where $U^\dagger = U^T \left(U U^T\right)^{-1}$ is the right inverse of U. A similar reasoning applies to the other modes.

B. ALS for training

Since we have shown that f is quadratic with respect to blocks Δ_j, let us now derive formally an algorithm to estimate these blocs Δ_j using Alternating Least Squares, as well as $\mathcal{X}$. ALS is reported as a baseline algorithm for unconstrained PARAFAC [14]. It is also a well studied algorithm for computing the Tucker decomposition, a decomposition to which our training problem reduces to when $r = 1$ [6]. The proposed ALS is summarized in Algorithm 1. Factors Δ_j are estimated in an alternating fashion using (6), while $\mathcal{X}$ is computed using any sparse coding method. Little can be said about convergence of Algorithm 1, since ALS for the training problem does not have a local convergence guarantee [14], and the sparse coding step is not even guaranteed to decrease the cost function. Yet, no convergence problems have been observed in practice.

Special attention has to be devoted to satisfy the unit-norm constraint. Like most dictionary learning algorithms, we normalize the estimated $D = \sum_q D_{1,q} \boxtimes D_{2,q} \boxtimes D_{3,q}$ at each iteration. However, at line 17 of Algorithm 1, we store both the blocks $\{D_{j,q}\}_{q=1,\dots,r}^{j=1,\dots,3}$ and the norms Σ separately to make use of the Kronecker structure in the sparse coding step. Even if this ad-hoc normalization step may marginally increase the cost function, it was not observed to decisively impact the algorithm's performance.

IV. Complexity analysis and practical speedups

The advantages of HO-SuKro are threefold. First, as the number of parameters to estimate in D is smaller than in the usual dictionary learning framework, HO-SuKro is expected to better resist a diminution of sample size. This behavior has been observed in [2]. To be exact, the number of parameters in the Kronecker-structured D is $r(\sum_j m_j d_j)$ while the full D has $\prod_j m_j d_j$ entries. Therefore assymptotically this dimentionality reduction is significant if r is smaller than the product

Algorithm 1 Alternating Least Squares for HO-SuKro

1: **INPUTS:** Data $\mathcal{Y}$, initial $D_{j,q}$ ($j \in [1,2,3], q \in [1,\dots,r]$)
2: Write $D_{4,q} = I_n$ for all $q \leq r$
3: **while** stopping criterion is not met **do**
4: ▷ *Sparse Coding*
5: Solve $X_{[4]} = \operatorname{argmin}_X \|Y_{[4]}^T - DX\|_F + g(\mathcal{X})$ using any sparse coding algorithm.
6: ▷ *Dictionary Update*
7: **while** update on $D_{j,q}$ is significant **do**
8: **for** j from 1 to 3 **do**
9: Set $U_q = X_{[j]} \left(\boxtimes_{l \neq j} D_{l,q}\right)^T$ for all $q \leq r$
10: Set $U = [U_1; \dots; U_r]$ stacked vertically
11: $\Delta_j = Y_{[j]} U^T \left(UU^T\right)^{-1}$
12: Set $[D_{j,1}, \dots, D_{j,r}] = \Delta_j$ stacked horizontally
13: **end for**
14: **end while**
15: ▷ *Dictionary Column Normalization*
16: $\Sigma(k,k) = 1/\|D(:,k)\|_2$ for all $k \in \{1,\dots,d\}$
17: $D = (\sum_{q=1}^r D_{1,q} \boxtimes D_{2,q} \boxtimes D_{3,q})\Sigma$
18: **end while**
19: **OUTPUTS:** Estimated blocks $D_{j,q}$, norms Σ, sparse $\mathcal{X}$

of all but one product $m_j d_j$, which is a mild assumption. Second, for the same reasons, the storage of D is less costly.

Third, every time a product D with any collection of vectors $V = [v_1, \dots, v_n] \in \mathbb{R}^{d \times n}$ is computed, one may actually use the following result:

$$\left(\sum_{q=1}^{r} D_{1,q} \boxtimes D_{2,q} \boxtimes D_{3,q}\right) V = \sum_{q=1}^{r} \mathcal{V} \times_1 D_{1,q} \times_2 D_{2,q} \times_3 D_{3,q} \quad (7)$$

where $\mathcal{V} \in \mathbb{R}^{d_1 \times d_2 \times d_3 \times n}$ is a tensorized version of V and $\times_i$ is the mode-wise product [14] that verifies $A \times_i \mathcal{V} := AV_{[i]}$. So, it comes down to a sequence of mode products with the smaller matrices $D_{j,q}$ which involves approximately $r(\sum_j m_j)(\prod_j d_j)n$ term to term products instead of $(\prod_j m_j d_j)n$ for a full matrix product DV. This result will be particularly useful in two occasions: 1) Dictionary Update: calculating U_q for all $q \leq r$ (line 9 in Alg. 1); 2) Sparse coding (line 5 in Alg. 1): discussed in Section IV-B.

A. Dictionary update step

Table I lists the computational complexities for one iteration of the inner loop in the dictionary update step (lines 8-13). The listed operations are performed for each of the modes and repeated a certain number of times (outer loop, lines 7-14). Our experiments indicate that very few outer iterations suffice to provide good convergence (typically less than five).

TABLE I: Computational complexity: dictionary update

Operation	Complexity
$U_q,\ \forall q \leq r$	$r(\sum_{l \neq j} m_l)(\prod_l d_l)n$
$Y_{[j]}U^T$	$r(\prod_l m_l)d_j n$
UU^T	$r^2(\prod_{l \neq j} m_l)d_j^2 n$
$(UU^T)^{-1}$	$(rd_l)^3$

TABLE II: Time spent in operation $D^T\rho$ for 10 iterations

m	[6, 6, 6]	[8, 8, 8]	[10, 10, 10]
d	[12, 12, 12]	[16, 16, 16]	[20, 20, 20]
OMP Cholesky	46 %	83 %	93 %

Considering that typically $n \gg \prod_j d_j > \prod_j m_j \gg r$, the first and the third operations are the most costly. Nevertheless, thanks to the tensor structure we manage to completely avoid complexities scaling with $(\prod_l m_l)(\prod_l d_l)n$ as it would be the case for a single product DX or D^TY with an unstructured dictionary. Whenever multiple mode-products are to be performed, the mode ordering can be chosen wisely to minimize the total complexity.

B. Structured sparse coding

Sparse coding algorithms, either greedy (like OMP and its variants) or convex-relaxation-based (like Iterative Soft-Thresholding algorithm – ISTA – and its variants), can also profit from the proposed structure in the dictionary. Basically, any product with the dictionary can be replaced by a mode-product with the smaller factors $D_{j,q}$ as shown in eq. (7).

In OMP-like algorithms, the complexity of an iteration is dominated (see Table II for some empirical evidence) by the calculation of the inner product between the dictionary and the current residual in $\mathcal{O}(\prod_j m_j d_j)$. For an HO-SuKro dictionary, this operation can be accelerated using (7) to a complexity of $\mathcal{O}(r(\sum_j m_j)(\prod_j d_j))$. This operation is followed by a normalization using Σ with a minor computation cost.

The mentioned accelerations can be used to speedup the learning process (sparse coding step on line 5 of Alg. 1) and also after it, once the dictionary is learned and applied repeatedly to some targeted data. In considered application (patch-based Hyperspectral image denoising), the latter corresponds to the denoising itself, using the learned dictionary.

Table II shows the percentage of time spent in the matrix-vector product $D^T\rho_t$ between the dictionary and the current residual, $\rho_t = y - Dx_t$ given the estimated coefficient vector x_t at iteration t, for a standard implementation of OMP with an unstructured dictionary. A higher time percentage implies a higher potential of speedup with the structured dictionary. Note that the percentage grows with the problem's dimensions.

Other more intricate implementations of OMP, such as Batch-OMP [15], which assumes the precomputation of both D^TY and the Gram matrix (D^TD), still admit some acceleration with the proposed structure. This, however, requires a more involved discussion, which the authors have put aside for future works.

C. From theoretical to practical speedups

The proposed accelerations, in both dictionary update and sparse coding steps, rely on the theoretical complexity of tensor mode-products. In this section, we evaluate to which extent such theoretical gains translate into actual speedups.

As explained, a product with the HO-SuKro dictionary becomes a sequence of mode-products. When implementing such

TABLE III: Speedups in matrix-vector products

Dimensions		Theoretical speedup | *Empirical speedup*					
m	d	$r=1$		$r=3$		$r=5$	
[6, 6, 6]	[12, 12, 12]	20.3	*0.8*	6.8	*0.4*	4.9	*0.3*
[8, 8, 8]	[16, 16, 16]	36.6	*4.5*	12.2	*2.0*	7.3	*1.4*
[10, 10, 10]	[20, 20, 20]	57.1	*13.2*	19.1	*5.7*	11.4	*3.5*

operations, it is necessary to repeatedly unfold the data tensor along each of its modes. Given a tensor $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3 \times n}$, the mode-$j$ unfolding is obtained by cascading two operations: (i) a permutation of the indexes of $\mathcal{X}$ so that mode j becomes the first dimension of the permuted tensor, and (ii) a reshape operation that unfolds all modes but the first along the second dimension. Although such operations are neglected in theory (considered $\mathcal{O}(1)$), this is not the case in practice as a reorganization of the tensor entries in memory might be required.

This seemingly harmless overhead actually creates a considerable gap between theoretical and empirical speedups, especially for smaller dimensions, as exemplified in Table III. Nevertheless, some considerable acceleration is still achieved in practice, in higher dimensions. A lower-level implementation [16] can be considered to try and tighten this gap, which would be of great interest since there is still a large acceleration potential to be exploited.

V. Experiments

A. Hyperspectral image denoising

We now evaluate the proposed Algorithm 1 in a Hyperspectral image (HSI) denoising task. The correlation along spectral bands in HSI have been found very useful to improve HSI denoising, as opposed to conventional denoising techniques based solely on 2D modeling. As a result, modern HSI denoising techniques have evolved to incorporate spectral information. Our structured dictionary framework allows us to manipulated directly the 3D data and properly exploit its original structure. Patch-based approaches, like ours, have rarely been explored in the literature so far. For a more extensive survey of this domain, we refer the reader to [17].

We will initially compare our method to other techniques based on the sparse modeling assumption: fixed sparsifying transforms (2D and 3D [18] Wavelet) and patch-based unstructured learned dictionaries (K-SVD [15]). We will see that we manage to outperform both of them. In the Hyperspectral imaging literature, another assumption besides sparsity proved to be very useful: the low-rankness of the image itself. Since we don't take this assumption into account, we don't manage to match the performance of state-of-the-art methods which combine both sparsity and low-rank assumptions. We still manage, nevertheless, to approach their performance, which is quite encouraging. The integration of the low-rank assumption in our model is envisioned for future work.

B. Simulation setup: a patch-based approach

Given a hyperspectral image corrupted with random Gaussian noise with standard deviation σ uniform over all spectral modes, we collect n 3D-patches with dimensions $\{m_1, m_2, m_3\}$ from the noisy image to form our training data $Y \in \mathbb{R}^{m_1 m_2 m_3 \times n}$. As a pre-treatment, each patch is re-centralized to have a zero mean along its pixel values. The patches are taken uniformly-spaced and partially overlapping.

A dictionary is learned from this data with 20 iterations of alternating optimization. Sparse coding is performed by OMP with an error threshold τ proportional to the noise level σ: $\tau = \sigma\sqrt{\prod_j m_j}$. The learned dictionary is then used to reconstruct patches with a one-pixel step. The recovered patches are averaged in the overlapping pixels along with the noisy image itself to form the final denoised image.

Two standard HSI images, from *San Diego* and *Houston* datasets, were cropped to $256 \times 256 \times 100$ pixels and used in the experiments. Reported results were obtained using 100000 training samples, which corresponds roughly to a 4-pixel step between adjacent patches. Although some performance improvement was observed when increasing this number, it does not compensate for the resulting raise in training time. Dictionaries were initialized with a 3D-DCT. In HO-SuKro's $r > 1$ case, the remaining terms were initialized with unit-norm Gaussian random matrices, and revealed to be quite robust to the initialization. The ALS loop (line 7 in Alg. 1) was carried over until the update on the blocks $D_{j,q}$ fell below a threshold empirically set to 10^{-1}, i.e.: $\frac{1}{r\sqrt{d}} \sum_q \|D_{j,q}^{\text{old}} - D_{j,q}^{\text{new}}\|_F < 10^{-1},\ \forall j$.

TABLE IV: Output SNR for various patch sizes – San Diego

Algorithm	Patch size		Input SNR			
	m	d	10	15	20	25
K-SVD	[6, 6, 6]	[12,12,12]	21.77	25.31	29.09	32.68
	[8, 8, 8]	[16,16,16]	21.51	25.19	29.2	32.95
	[10,10,10]	[20,20,20]	21.52	25.34	29.42	33.24
	[6, 6,20]	[12,12,20]	22.49	26.06	29.91	33.21
HO-SuKro ($r = 3$)	[6, 6, 6]	[12,12,12]	22.05	25.35	28.9	32.3
	[8, 8, 8]	[16,16,16]	22.73	25.82	29.22	32.64
	[10,10,10]	[20,20,20]	23.08	26.35	29.73	**33.27**
	[6, 6,20]	[12,12,20]	**24.10**	**27.07**	**30.09**	33.22

C. Denoising performance

Table IV shows the denoising performance of the proposed structured dictionaries compared to K-SVD as an unstructured counterpart. In order not to overburden the analysis, we report only the HO-SuKro results with rank $r=3$, which we judged to represent a good performance-complexity compromise.

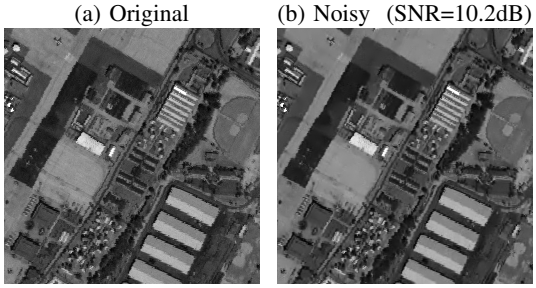
Increasing the patch size improves significantly the performance of HO-SuKro, while that of K-SVD doesn't benefit as much and may even deteriorate, indicating the onset of overfitting. This can be attributed to the growing number of parameters to estimate (as the dictionary size grows) making the number of available training data insufficient. HO-SuKro, thanks to its structured nature, avoids this issue.

More than simply increasing the patch size, what was empirically observed to drastically improve performance was increasing the patch dimension in the spectral mode, as shown in the last line of Table IV. Naturally, bigger patches also imply higher learning and denoising times. It is, thus, a compromise to be considered according to the available resources.

TABLE V: Output SNR [dB] comparison with literature

Image	Algorithm	Input SNR [dB]			
		10	15	20	25
San Diego	Wavelet 2D	14.75	18.00	21.70	25.92
	Wavelet 3D	23.11	26.04	28.91	31.68
	FORPDN	22.23	24.17	26.42	29.00
	HyRes	**25.38**	**28.60**	**31.75**	**34.70**
	HO-SuKro	24.10	27.07	30.09	33.22
Houston	Wavelet 2D	14.22	17.67	21.43	25.80
	Wavelet 3D	22.35	25.54	28.65	31.86
	FORPDN	22.80	25.46	28.09	30.74
	HyRes	**26.00**	**29.35**	**33.24**	**37.05**
	HO-SuKro	23.29	26.63	29.93	33.20

(a) Original (b) Noisy (SNR=10.2dB)



(c) HyRes (25.4dB) (d) HO-SuKro (24dB)

Fig. 1: Example of denoised images (100^{th} spectral band)

Table V compares HO-SuKro's performance to that of other techniques from the literature. By directly applying the proposed structured dictionary learning algorithm to Hyperspectral data, without any further domain-specific adaptation, we already obtain results comparable to the literature. Both HO-SuKro and K-SVD outperform the wavelet-based approaches (2D and 3D [18]), corroborating the interest of learning the sparsifying dictionary from data. HO-SuKro also consistently outperforms FORPDN [19] which exploits the correlation within spectral bands. Naturally, we don't manage to reach state-of-the-art performance (HyRes [20]) for this task, which makes use of a meaningful low-rank prior on the HSI. The performance gap is around 1.5dB and 3dB for *San Diego* and *Houston* images respectively. An example of denoised image is provided in Figure 1. A closer look reveals that our approach may over-smooth some details.

VI. Conclusion

An alternate minimization algorithm was proposed in this paper for learning tensor-structured dictionaries designed to tackle the inherent multidimensional structure of the data. We analyzed the computational complexity of the algorithm and made some considerations on the expected empirical speedups.

Hyperspectral image denoising experiments were performed and the tensor structure constraint on the dictionary proved to be beneficial in terms of performance compared to a completely unstructured dictionary. Some adaptations are envisioned to improve our performance in this specific task and try to reach the state-of-the-art, notably the incorporation of the low-rank constraint on the data.

References

[1] Ron Rubinstein, Alfred M Bruckstein, and Michael Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[2] Cassio F. Dantas, Jérémy E Cohen, and Rémi Gribonval, "Learning fast dictionaries for sparse representations using low-rank tensor decompositions," in *LVA/ICA*, Guildford, UK, 2018.

[3] Jeremy E Cohen, "About notations in multiway array processing," *arXiv preprint arXiv:1511.01306*, 2015.

[4] S. Hawe, M. Seibert, and M. Kleinsteuber, "Separable dictionary learning," in *Proceedings of the IEEE CVPR*, 2013, pp. 438–445.

[5] Y. Peng, D. Meng, Z. Xu, C. Gao, Y. Yang, and B. Zhang, "Decomposable nonlocal tensor dictionary learning for multispectral image denoising," in *IEEE CVPR*, 2014, pp. 2949–2956.

[6] Cesar F. Caiafa and Andrzej Cichocki, "Multidimensional compressed sensing and their applications," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3, no. 6, pp. 355–380, 2013.

[7] M. Ghassemi, Z. Shakeri, A. D. Sarwate, and W. U. Bajwa, "STARK: Structured Dictionary Learning Through Rank-one Tensor Recovery," in *IEEE CAMSAP*, 2017.

[8] T. Tsiligkaridis and A. O. Hero, "Covariance Estimation in High Dimensions Via Kronecker Product Expansions," *IEEE Transactions on Signal Processing*, vol. 61, no. 21, pp. 5347–5360, 2013.

[9] Fetsje Bijma, Jan De Munck, and Rob M Heethaar, "The spatiotemporal MEG covariance matrix modeled as a sum of Kronecker products," *NeuroImage*, vol. 27, pp. 402–15, 2005.

[10] C. F. Dantas, M. N. da Costa, and R. R. Lopes, "Learning Dictionaries as a sum of Kronecker products," *IEEE Signal Processing Letters*, 2017.

[11] Kim Batselier and Ngai Wong, "A constructive arbitrary-degree kronecker product decomposition of tensors," *Numerical Linear Algebra with Applications*, 2017.

[12] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE, 1993, pp. 40–44.

[13] Amir Beck and Marc Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[14] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Sept. 2009.

[15] Ron Rubinstein, Michael Zibulevsky, and Michael Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," *CS Technion*, vol. 40, no. 8, pp. 1–15, 2008.

[16] Devin A. Matthews, "High-performance tensor contraction without transposition," *SIAM J. Scientific Computing*, vol. 40, no. 1, 2018.

[17] Behnood Rasti, Paul Scheunders, Pedram Ghamisi, Giorgio Licciardi, and Jocelyn Chanussot, "Noise reduction in hyperspectral imagery: Overview and application," *Remote Sensing*, vol. 10, no. 3, 2018.

[18] Abdullah A Basuhail and Samuel Peter Kozaitis, "Wavelet-based noise reduction in multispectral imagery," in *Algorithms for multispectral and hyperspectral imagery IV*. International Society for Optics and Photonics, 1998, vol. 3372, pp. 234–241.

[19] B. Rasti, J.R. Sveinsson, M.O. Ulfarsson, and J.A. Benediktsson, "Hyperspectral image denoising using first order spectral roughness penalty in wavelet domain," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2458–2467, 2014.

[20] B. Rasti, M. O. Ulfarsson, and P. Ghamisi, "Automatic hyperspectral image restoration using sparse and low-rank modeling," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 12, pp. 2335–2339, Dec 2017.